The devil is in discretization discrepancy. Robustifying Differentiable NAS with Single-Stage Searching Protocol

Konstanty Subbotko* Wojciech Jablonski* Piotr Bilinski
University of Warsaw

Abstract

Neural Architecture Search (NAS) has been widely adopted to design neural networks for various computer vision tasks. One of its most promising subdomains is differentiable NAS (DNAS), where the optimal architecture is found in a differentiable manner. However, gradient-based methods suffer from the discretization error, which can severely damage the process of obtaining the final architecture. In our work, we first study the risk of discretization error and show how it affects an unregularized supernet. Then, we present that penalizing high entropy, a common technique of architecture regularization, can hinder the supernet's performance. Therefore, to robustify the DNAS framework, we introduce a novel single-stage searching protocol, which is not reliant on decoding a continuous architecture. Our results demonstrate that this approach outperforms other DNAS methods by achieving 75.3% *in the searching stage on the Cityscapes validation dataset and attains performance* 1.1% *higher than the optimal network of DCNAS on the non-dense search space comprising short connections. The entire training process takes only 5.5 GPU days due to the weight reuse, and yields a computationally efficient architecture. Additionally, we propose a new dataset split procedure, which substantially improves results and prevents architecture degeneration in DARTS.*

1. Introduction

Neural architecture search (NAS) is a field that automates the designing of neural networks. Differentiable neural architecture search (DNAS) denotes the set of gradient-based NAS techniques. In these methods, we relax the discrete architecture space into the space of continuous architectures [23] and optimize it using stochastic gradient descent.

DNAS framework can be decomposed into three stages:

1. the *searching* stage, where a "supernet" assembling all architecture candidates as subnetworks is trained,
2. the *decoding* stage, which retrieves a discrete architecture from a continuous search space, and
3. the *retraining* stage, where a retrieved architecture is trained for a longer time and with newly initialized weights.

The usage of supernet greatly reduces computational costs by enabling weight-sharing across a vast number of different architectures [23, 25]. However, despite its computational effectiveness and significant potential, practical applications of DNAS are hindered by the severe fragility and instability [9, 19, 40]. One of the major issues, which we refer to as the *discretization error*, concerns the poor architecture optimization process and emerges at the decoding stage during the discretization procedure [2, 9, 30, 40]. Discarding operations or connections can yield substantially different architecture when a supernet is poorly discretized and has a high entropy at the end of the training. As a result, it can impact the searching-retraining correlation. Thus, even a network retrieved from a well-performing supernet might underperform after retraining.

In this work, we shed more light on the discretization error and propose a novel solution to address it. Contrary to

Method	Parameters	Entropy	
		Start	End
Auto-DeepLab [22]	Edges	0.259	0.256
	Operations	0.260	0.258
DARTS [23]	Topology	0.126	0.126
	Operations	0.260	0.256
Ours	Edges	0.128	0.127
	Operations	0.346	0.345

Table 1. Lack of implicit entropy regularization in a supernet. The average entropy of architectural parameters across different methods and different tasks at the start and the end of the searching stage. For DARTS, we adopt decoupled topology search [14], which introduces a new set of parameters.

*Work done while being at University of Warsaw

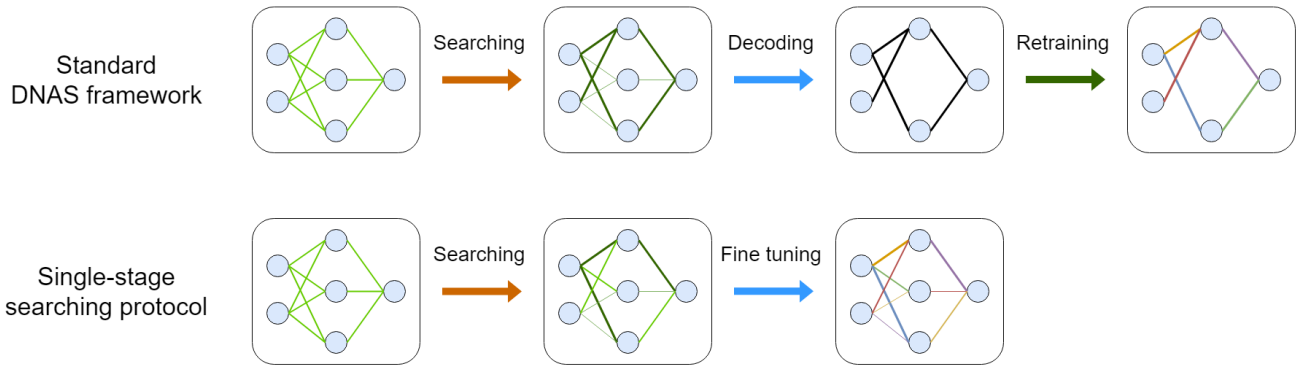


Figure 1. Illustration of the single-stage searching protocol. We replace both the decoding and the retraining stages with a new fine-tuning phase, during which architecture is frozen. By reusing weights, we save a considerable amount of the retraining time. We keep the optimized architectural parameters in the final network, which means that edges in a supernet take on real values, unlike in the standard DNAS framework.

the common approach, we perform experiments in the semantic segmentation task [11] for two reasons. First, our approach is better suited for tasks that can benefit from dense architectures and a certain design of the search space. Second, we find it more effective in highlighting some problems than the extensively studied task of image classification, where differences between approaches can even be statistically insignificant. Our experiments in Tab. 1 reveal a lack of implicit entropy regularization in the vanilla DNAS framework. We can observe that the average entropy of architectural parameters remains constant throughout the training, indicating that a considerable number of operations in a supernet contributes to the prediction at the end of the searching. This might in turn cause the discretization error.

One of the common approaches to tackle the discretization error is to impose a one-hot distribution over architectural parameters by regularizing a supernet [9, 30, 42]. In our work, we take a closer look at the method of penalizing high entropy and highlight its shortcomings. As we demonstrate in Sec. 4, such a regularization induces a trade-off between discretization and obtained results. Namely, we show that the magnitude of the entropy loss negatively correlates with the discretization error, which indeed suggests a need for a strong regularization. At the same time, our experiments indicate that it can degrade supernet's performance in the searching stage. We also consider another variant of dynamic entropy loss regularization [30], which alleviates the performance issue, but does not eliminate it entirely.

To this end, we propose to approach the problem from a different angle and to train the supernet in a fully proxyless manner by introducing a *single-stage* searching protocol. The approach is illustrated in Fig. 1. We simplify the searching process by replacing both the decoding and the retraining stages with a new fine-tuning phase on the top of the searching stage. We do not perform discretization and, thereby, we treat optimized supernet with all its trained parameters as the final model. By reusing weights from the searching stage, we considerably reduce the total training time. In this approach, it is crucial to design a search space that is both expressive and computationally efficient. For that reason, our method might not be yet appropriate for certain tasks. We apply our single-stage searching method to the non-dense DCNAS search space [42], which includes transmissions spanning only between consecutive layers. As we show in Sec. 4, our approach is on par with, or even surpasses, other state-of-the-art DNAS models in terms of computational requirements.

The single-stage searching protocol also addresses another deficiency of the DNAS framework, which is often overlooked - prohibitively high computational complexity. Because of the proxy searching procedure, each retrieved architecture is trained from scratch, which imposes extra costs. Furthermore, the DNAS method can suffer from a poor searching-retraining correlation [42]. As a result, in addition to the costs of finding the optimal network, several candidate architectures must be evaluated to counteract low correlation. Instead, our single-stage searching algorithm takes only 5.5 GPU days to converge by using a single set of weights throughout the training.

We validate our improvements on the Cityscapes dataset [11]. Our single-stage method outperforms other DNAS methods in the searching stage by achieving 75.3% on the validation set. Moreover, it attains performance 1.1% higher than the final derived network of DCNAS on the non-dense search space, which demonstrates the viability of the single-stage approach. In our experiments, we also perform an ablation study on a dataset split procedure [22, 23],

which shows that more optimal data usage yields up to 5.4% boost. Additionally, we demonstrate that it can prevent architecture degeneration in DARTS.

We summarize our main contributions as follows:

- We investigate the discretization error in a semantic segmentation task, empirically show its negative consequences, depending on the strength of the regularization.
- We study the entropy architecture regularization and demonstrate that it hinders the supernet's performance.
- We introduce a fully proxyless, single-stage searching protocol that eliminates the discretization error by thoroughly fine-tuning the supernet. We demonstrate that it yields a computationally efficient architecture, which outperforms DCNAS on a comparable search space. Also, we show its superiority in terms of the total training time.
- We conduct an ablation study on the training dataset split approach in DNAS methods, highlighting a much more optimal dataset usage, which considerably enhances performance and prevents architecture degeneration in DARTS.

2. Related work

Neural architecture search is a collection of novel techniques aiming to automate the neural network design process. It can automatically discover the optimal neural network architecture for a given task or dataset. NAS research concerns primarily evolutionary [27, 28, 37], reinforcement learning [1, 29, 45–47] and DNAS [15, 23, 31, 33, 35] methods.

In Differentiable NAS (DNAS), the optimal architecture can be found using stochastic gradient descent, thanks to the differentiable representation of the search space [23]. Each architectural choice is assigned a continuous parameter. The search space is represented as a large, weight-sharing supernet, where different architecture candidates correspond to different subsets of this supernet. Searching over this search space essentially comes down to training the network. Afterward, the optimal discrete architecture is retrieved from a supernet and retrained from scratch for a longer time.

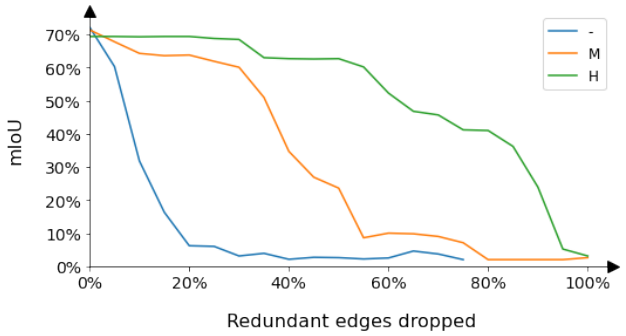
NAS algorithms search for a network within a predefined search space. The search space must be expressive enough to include a wide range of candidate architecture and also be efficiently optimizable. In NASNet [47], the network comprises a sequence of convolutional cells, all sharing the same architecture. The cell, composed of multiple blocks, forms the search space. The spatial dimension throughout the network is controlled manually. Several NAS applications to the image classification task follow the same design [21, 23, 29, 47].

AutoDeepLab [22] builds upon DARTS [23] and adapts its approach to semantic segmentation by introducing a significantly larger network-level hierarchical architecture search space, while operating on the same cell-level search space as DARTS. DCNAS [42] extends the idea of hierarchical search space and introduces a densely connected search space, incorporating long-range connections reaching every cell and every spatial level. DPC [4] proposes a recursive search space formed by a dense prediction cell, which uses a segmentation-specific set of operations, such as atrous convolution or pooling. In the searching stage, DPC constructs a proxy task of finding the dense prediction cell on the top of the backbone pretrained on ImageNet [12].

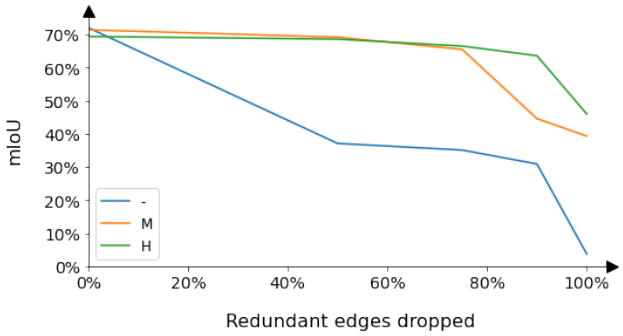
Discretization issue. Several works derived from DARTS [23] focus on the searching to retraining transition. Much attention was drawn to the collapsing phenomenon in DARTS, where a supernet assigns excessive weights to skip connections [7, 9, 10, 19, 38, 40]. RobustDARTS [40] alleviates this issue through a hand-crafted early-stopping strategy. SmoothDARTS [7] enhances architecture generalization by perturbing architecture parameters, thus smoothing the loss landscape. Some other works improve DARTS decoding efficacy by better estimating the importance of operations [32, 41].

The concept of discretization issue is established in the literature [2, 9, 30, 40]. Fair DARTS [9] observes the discretization discrepancy by visualizing softmax activations and introduces a zero-one loss, which imposes a one-hot distribution. GOLD-NAS [2] addresses the issue by using hardware constraints to penalize significant architectural parameters and progressively prune weak operations. The most related to our work in terms of discretization study is DA^{2}S [30]. The authors show that vanilla DARTS suffers from a performance collapse by performing inference on a discretized supernet. To alleviate this collapse, they introduce dynamic entropy loss to impose one-hot distribution in the later stages of the training. DCNAS [42] likewise regularizes architectural parameters to diminish insignificant transmissions.

Proxyless searching involves sharing the training protocol between a supernet and a retrained network. In particular, this implies using the same hyperparameters, such as batch size or image crop. ProxylessNAS [3] samples paths within the search space to facilitate proxyless searching. Similarly, DCNAS [42] probes candidate architectures by sampling connections. This approach, along with masking a subset of channels in each cell [36, 42], makes proxyless searching viable. Another study [8] divides the searching stage into a few phases and performs a stepwise discretization. This can be considered a related work to proxyless searching, as the proxy gradually decreases. In our work, we extend the idea of proxyless searching and propose to optimize a supernet in an end-to-end manner using target hyparameters. However, unlike other approaches, we do not retrain it. Instead, we reuse the already trained weights, thus significantly reducing computational requirements.



(a) Discretiation w/o fine-tuning



(b) Discretiation with fine-tuning

Figure 2. Visualization of the discretization error across different entropy loss magnitudes. For more details, see Sec. 4.2.

3. Methods

In this section, we first present cell-level and network-level architecture search space. Subsequently, we introduce the single-stage searching protocol, which can save computational costs and eliminate the discretization error, followed by a description of the entropy loss.

3.1. Search space

Network-level architecture. The supernet comprises three modules: the *stem*, the *backbone*, and the *decoder*. Following DCNAS [42], we utilize multi-scale feature representation in our network. We adopt the stem module used by Auto-DeepLab and adjust it to the multi-scale network structure by performing interpolations of an input image. For the decoder, we reuse the prediction head designed by DCNAS.

A dense rectangular-like grid of cells forms the backbone. Each cell corresponds to a particular layer and resolution. Resolutions reach up to a downsampling rate of 32. Network-level transmissions span between adjacent cells in consecutive layers. We assign an architectural parameter $\beta^l_{s' \to s}$ to a transition in layer l between resolution s' and s. We define an input to a cell as a weighted average over the outputs of its predecessors:

$$X^l_s = \sum_{t \in \{s/2,\, s,\, 2s\}} \mathcal{P}\big(Y^{l-1}_t\big)\ \hat{\beta}^l_{t \to s}\,. \tag{1}$$

Here, X^i_j and Y^i_j denote the input and output of an j-th cell in a i-th layer, respectively. $\mathcal{P}$ is a shape-aligning preprocessing operation applied separately to each feature-maps. $\hat{\beta}$ are normalized scalars, indicating edge relative importance within a cell:

$$\hat{\beta}^l_{s' \to s} = \frac{\exp(\beta^l_{s' \to s})}{\sum_{t \in \{s/2,\, s,\, 2s\}} \exp(\beta^l_{t \to s})}\,. \tag{2}$$

Cell-level architecture. We follow DCNAS and use an inverted bottleneck [17] cell structure similar to [3]. Operator space consists of convolutions with different kernel sizes. We assign a parameter $\alpha^k_{s,l}$ to a block with convolution o_k with a kernel of size k in each cell. The output of a cell is defined as follows:

$$Y^l_s = \sum_k \hat{\alpha}^k_{s,l}\ o_k\big(X^l_s\big)\,, \tag{3}$$

where $\hat{\alpha}^k_{s,l}$ are parameters normalized using softmax, analogously to network-level transmissions. In certain experiments, we adopt the channel sampling scheme [36, 42], which reduces the number of feature-maps filters before processing them in a cell.

3.2. Single-stage searching protocol

We propose to train the network in a fully proxyless way by introducing the single-stage searching protocol. We shrink the DNAS framework by dropping the decoding and retraining stages, retaining only the searching stage, which comprises three phases:

1. *warmup* phase, which precedes architecture optimization, and with gradient updates performed exclusively on weights,
2. *searching* phase, where architecture and weights are jointly optimized,
3. *fine-tuning*, in which architecture is fixed, and only the weights are optimized.

The warmup phase has been introduced to prevent architecture from degeneration caused by using randomly initialized weights [22]. In the searching phase, we alternately update architectural parameters and weights by adopting the following bilevel optimization scheme:

- Update network weights w by $\nabla_w \mathcal{L}_A(w, \alpha, \beta)$,
- Update architecture α, β by $\nabla_{\alpha,\beta} \mathcal{L}_B(w, \alpha, \beta)$,

where $\mathcal{L}_A$ and $\mathcal{L}_B$ denote cross-entropy loss computed on two subsets of training data A and B, respectively. We also include the entropy regularization term in $\mathcal{L}_B$.

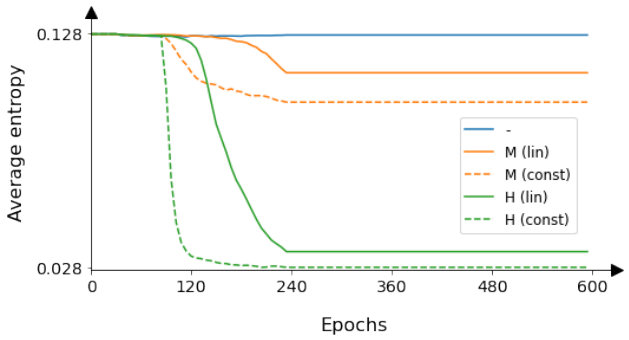


Figure 3. The average entropy of architectural parameters throughout the training. Dashed and solid curves correspond to supernets trained with the constant and the linear entropy scaling function, as described in Sec. 3.3. Curves denoted by -, M, and H refer to supernets trained without entropy loss, with medium entropy loss, and with high entropy loss, respectively.

The fine-tuning phase can be perceived as a replacement for the retraining stage. However, we managed to reduce the overall training time considerably. Unlike in the standard DNAS framework, we do not perform decoding, and thus, we can efficiently reuse weights that have already been trained in the preceding phases. In Sec. 4, we show that this approach can achieve optimal results.

3.3. Entropy loss

Similar to other works [30, 42], we use the entropy loss term in our experiments to regularize architectural parameters. The purpose is to penalize the excessive usage of insignificant transmissions and operations in a supernet. We formulate the term in the following way:

$$\mathcal{L}_{\text{ent}} = c_\beta \; f(t) \sum_{\hat{\beta}_i} \hat{\beta}_i \, \ln \hat{\beta}_i \,. \tag{4}$$

Here, $f(t)$ is a scaling term dependent on the time t, and c_β denotes the overall entropy loss magnitude for the network-level architectural parameters. We apply the same regularization to the cell-level parameters c_α.

In our experiments, we consider two variants of the scaling function. The *Linear* function refers to a linear scaling term, which gradually increases the magnitude of the entropy loss from 0 to 1. The *Constant* function sets $f(t) = 1$ and corresponds to the default approach of applying the entropy loss.

4. Experiments

We validate our methods on Cityscapes [11], a go-to semantic segmentation dataset for evaluating the searching efficacy of DNAS. It enables us to directly compare our approach with other methods in a challenging environment. The dataset consists of high-resolution images with fine and coarse annotations. The former are split into sets of 2975, 500, and 1525 images for training, validation, and testing, respectively. The latter provides labels for 20000 training samples, but details and object boundaries are coarsely annotated.

4.1. Implementation details

The large model takes 1.4 days to train for 600 epochs on 4 Tesla V100 32GB GPUs. We use syncBN [24] to synchronize statistics in the batch normalization layers across devices. Respectively, we assign 5%, 35%, and 60% of epochs to the warmup, the searching, and the fine-tuning phases. We apply architecture regularization after 15% of epochs. However, in our experiments, the supernet is not excessively sensitive to changes in these hyperparameters.

Following previous works [22, 42], we use two optimization strategies to train architectural parameters and operation weights. For the former, we adopt Adam [18] with a learning rate of 0.003 and weight decay of 0.001. For the latter, we use SGD parameterized by a learning rate of 0.003 and weight decay of 0.0005. As a data augmentation, we apply horizontal flipping, random scaling, color jittering, and random Gaussian noise. We set the batch size to 16 and train the network using crops of 512 ×1024. More details can be found in our implementation, which we release with the code.

We present different variants of models in Tab. 3. They vary in the number of layers L, the filter multiplier F, the expansion ratio Exp in the inverted bottleneck, and the channel sampling ratio S. Due to limited computational resources, we use the small model in the experiments concerning discretization error and entropy loss.

4.2. Emergence of discretization error

Fig. 3 illustrates the average entropy of c_β. We can observe a sharp drop in entropy for the step scaling function, which can negatively impact training dynamics, especially at higher magnitudes. This observation aligns with our results presented in Sec. 4.3. The experiment also shows that the unregularized supernet, denoted by a blue line, has severely non-discretized architectural parameters, which could lead to the discretization error.

To study more thoroughly how discretization is impacted by entropy regularization, we gradually discretize supernets trained with different magnitudes of entropy loss. Specifically, we perform inference after dropping a certain number of redundant edges based on the strength of their architectural parameters. It is important to note that while this can effectively measure the relative importance of an edge within a cell, it might not optimally rank edges according to their relevance across different cells in different parts of the

Function	Entropy magnitude			
	-	L	M	H
Constant	**70.9** $\pm$ 1.1%	70.2 $\pm$ 1.7%	69.6 $\pm$ 0.9%	68.1 $\pm$ 1.2%
Linear	**70.9** $\pm$ 1.1%	**70.9** $\pm$ 0.4%	70.8 $\pm$ 0.7%	68.8 $\pm$ 0.5%

Table 2. Results for different scaling functions and magnitudes averaged over three runs. *Function* denotes the time-dependent scaling term introduced in Sec. 3.3. *Entropy magnitude* refers to the different values of c_α and c_β. *L*, *M*, *H* denote low, medium and large magnitudes, respectively.

Name	L	F	Exp	S	FLOPs	Params
Small	10	16	3	1	57.7G	3.2M
-	14	16	6	1	109.4G	10.7M
Medium	14	64	6	1/4	380.1G	22.3M
Large	10	64	3	1	558G	47.3M

Table 3. Comparison of different models varying in size. See Sec. 4.1 for reference.

Method	FLOPs (G)	s-mIoU	t-mIoU
Auto-DeepLab	695	34.9%	80.3%
DCNAS	294.6	69.9%	**81.2**%
DCNAS (non-dense)	-	51.7%	73.3%
DPC	684	-	80.9%
Ours (Small)	57.7	71.4%	n/a
Ours (Medium)	380.1	74.4%	n/a
Ours (Large)	558	**75.3**%	n/a

Table 4. Comparison between different methods on the Cityscapes validation dataset. FLOPs are computed for the final networks and taken from DCNAS. In our case, we evaluate the performance of the supernet.

supernet. Nevertheless, we use it as a reasonable approximation.

The results are illustrated in Fig. 2a. First, we observe a quick collapse of a standard unregularized DNAS supernet after dropping merely 20% of its edges. Second, we empirically demonstrate a correlation between the strength of regularization and resistance to discretization. In particular, pruning 30% of the transmissions in a heavily regularized supernet does not result in any noticeable drop in performance.

We conduct the same experiments, but this time they are followed with a short post-decoding fine-tuning, which adapts transferred weights to a discretized architecture. The aim is to investigate whether discretized architecture lies in the neighborhood of the optimal architecture in parameter space. In such a scenario, performing a relatively small number of updates could retrieve optimal performance. Our findings, presented in Fig. 2b, confirm that a strong entropy regularization can effectively address the discretization issue. Conversely, a vanilla DNAS supernet generates a qualitatively different architecture, partially explaining the low correlation observed by DCNAS [42].

4.3. Negative impact of entropy loss

As we highlight in Sec. 1 and Sec. 2, the entropy loss term is an established solution to the discretization error in the literature. However, it causes a sudden drop in the entropy of architectural parameters, as illustrated in Fig. 3. This might lead to a suboptimal convergence of the supernet. Dynamic regularization results in a more gradual entropy decrease.

We study the efficacy of the dynamic and static entropy losses on the Cityscapes validation set. Experiments are conducted with different scaling functions and magnitudes of the entropy loss (c_α and c_β). We report an average mIoU over three runs to obtain accurate estimates. Results are shown in Tab. 2. We observe a consistent drop in performance for higher entropy magnitudes, indicating the emergence of the discretization-exploration trade-off. A linearly scaled entropy loss term alleviates the issue and outperforms the default regularization technique, albeit converges poorly for higher entropy magnitude, emphasizing the necessity for a more robust approach.

4.4. Proxyless searching

We validate the single-stage searching protocol, our remedy for the discretization issue, in Tab. 4. Namely, we report the mIoU of our approach and the state-of-the-art DNAS methods on the validation set. For all models, we also provide the number of floating-point operations. Our medium and large models outperform supernets of Auto-DeepLab and DCNAS in the searching stage by achieving 74.4% and 75.3%, respectively. The large model matches Auto-DeepLab and DPC in the number of floating-point operations, whereas the medium network requires as little as 30% more than

Method	Val	Coarse	ImageNet	Results
GridNet [13]				69.5
FRRN-B [26]				71.8
Ours (Large)				74.0
DCNAS [42]				82.8
PSPNet [44]	✓	✓	✓	81.2
DeepLabv3+ [5]	✓	✓	✓	81.3
HRNetV2 + OCR [39]	✓		✓	83.9
SparseMask [34]			✓	68.6
CAS [43]	✓	✓	✓	72.3
GAS [20]	✓	✓	✓	73.5
Auto-DeepLab [22]	✓			80.4
Auto-DeepLab [22]	✓	✓		82.1
RSPNet [6]	✓		✓	81.4
DPC [4]	✓	✓	✓	82.7
DCNAS [42]	✓	✓		83.6
DCNAS + ASPP [42]	✓	✓		84.3

Table 5. Cityscapes test set results. **Val**: Models are also trained using annotations from the validation set. **Coarse**: Models exploit coarse annotations. **ImageNet**: Models pretrained on ImageNet.

Method	Searching		Retraining
	Epochs	GPU (days)	Epochs
Auto-DeepLab	40	3	16000
DCNAS	120	5.6	800
DPC	28k × 80	2600	500
Ours (Large)	600	5.5	-

Table 6. Comparison of DNAS methods on Cityscapes in time efficiency. The searching stage time for Auto-DeepLab and DPC is provided for P100, which considerably overstates the costs. We estimate DPC epochs based on values of its hyperparameters [4].

DCNAS.

Importantly, the medium model attains 1.1% higher mIoU compared to the non-dense variant of DCNAS, which is most comparable to ours in terms of the search space design. Given these results and the reasonable computational cost, the supernet trained in a proxyless way can be considered a viable drop-in replacement for the final network of the state-of-the-art DNAS algorithms. We hypothesize that incorporating long-range connections to the search space might further elevate the performance.

We present results for the test set in Tab. 5. However, we

Weights		Architecture		mIoU
Fine	Coarse	Fine	Coarse	
0.5	0	0.5	0	69.9%
1	0	1	0	**75.3**%
1	0	0	1	74.6%
0	1	1	0	61.8%
1	1	1	1	75.2%

Table 7. Results of the large model trained with different data splits on the validation set. **0**: Annotations are not used in training. **0.5**: Half of the annotations are used exclusively to optimize given parameters. **1**: All of the annotations are used. In case we use both fine and coarse annotations, we train the parameters using all the data and then fine-tune them with only fine labels.

do not further optimize performance. We employ the same training and inference procedures as used for the validation set, which may potentially understate the obtained results.

Regarding training time, we benchmark our approach against other DNAS works in Tab. 6. In the conventional searching-retraining procedure, only a small amount of time is dedicated to finding the optimal architecture [4, 22, 23, 42]. The bulk of computational resources are devoted to retraining the derived architecture, significantly increasing the costs of using other DNAS methods. The single-stage searching protocol eliminates the need for a time-consuming retraining stage, thus providing the optimal architecture in only 5.5 GPU days.

4.5. Optimal dataset split

DNAS methods commonly address the emerging bilevel optimization problem by training architectural parameters, $\{\alpha, \beta\}$, and operation weights, $\{\omega\}$, on two disjoint subsets of the training dataset. DARTS introduced this to prevent poor architecture generalization. MiLeNAS [16] argues that optimizing architectural parameters on the entire training dataset is optimal. Results from experiments with varied splits are presented in Tab. 7. We show that joint optimization using the same data yields the best results, provided that batches are sampled separately in each iteration for both sets of parameters. We also use two separate optimizers. Combining fine with coarse annotations does not offer significant benefits, potentially due to the supernet's undertraining. Surprisingly, performing architecture updates with batches consisting exclusively of coarse annotations matches the training performance obtained using other splits. Additionally, our supernet achieves superior results in the searching stage, even when using a less optimal data split than the one used by DCNAS.

To further verify our findings, we test if the new split-

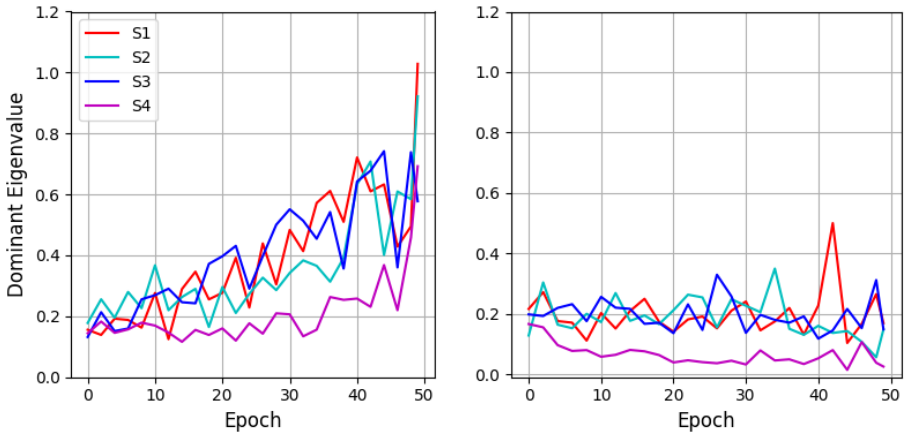


Figure 4. (left) dominant eigenvalues of $\nabla^2_\alpha \mathcal{L}_{valid}$ on four different search spaces with a dataset split; (right) dominant eigenvalues when searching on a single dataset. All experiments were conducted on CIFAR 10 dataset.

less dataset procedure affects architecture degeneration in DARTS [23]. RobustDARTS [40] showed that dominant eigenvalue of $\nabla^2_\alpha \mathcal{L}_{valid}$ is significantly increasing during searching and that there exists a correlation between large dominant eigenvalue and architecture degeneration across four different search spaces S1, S2, S3 and S4. First, we reproduce experiments presented in the paper for standard DARTS on CIFAR-10. Results are illustrated in Fig. 4 (left). We indeed observe that in all four cases dominant eigenvalues are steadily increasing and reach much larger values at the end of the searching. Second, we perform the same experiments, but with parameters optimized on the union of training and validation dataset. Fig. 4 (right) shows that this time in all four search spaces dominant eigenvalues are kept relatively small, which suggests that the network doesn't end up in a sharper local minima that would cause an architecture degeneration.

5. Conclusions

In this work, we conduct comprehensive experiments to investigate the discretization error. Our findings indicate that this issue emerges in an unregularized supernet during the searching stage. We study an established method for addressing this issue through entropy architecture regularization and highlight its shortcomings. As a remedy, we propose the single-stage searching protocol, a novel way of finding optimal neural networks using a gradient-based technique. Our method robustifies the DNAS framework by eliminating the discretization error that other methods suffer from. We show that it matches other state-of-the-art approaches in terms of performance and results within a similar search space. Also, we demonstrate that the joint optimization of architecture and weights on the full dataset yields better results, and prevents a well-known architecture degeneration phenomenon in DARTS. We believe that our work can be a starting point for creating even more powerful DNAS models. In future work, we aim to efficiently incorporate long-range connections into the search space to improve results even further.

Acknowledgements

We would like to express our gratitude to Nvidia and Google's TPU Research Cloud for making their compute resources available to us. Moreover, we would like to thank ICM UW and PLGrid for doing the same through the research grants. Neural Architecture Search is one of the most compute-intensive areas of deep learning, and this work could not have been possible without immense amounts of computing power.

References

[1] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. In *ICML*, 2017. 3

[2] Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. GOLD-NAS: gradual, one-level, differentiable. *CoRR*, abs/2007.03331, 2020. 1, 3

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3, 4

[4] Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jonathon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NIPS*, 2018. 3, 7

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 7

[6] Peihao Chen, Deng Huang, Dongliang He, Xiang Long, Runhao Zeng, Shilei Wen, Mingkui Tan, and Chuang Gan. Rspnet: Relative speed perception for unsupervised video representation learning. In *AAAI*, 2021. 7

[7] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 3

[8] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 3

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020. 1, 2, 3

[10] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: robustly stepping out of performance collapse without indicators. In *ICLR*, 2021. 3

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 5

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[13] Damien Fourure, Rémi Emonet, Élisa Fromont, Damien Muselet, Alain Trémeau, and Christian Wolf. Residual conv-deconv grid network for semantic segmentation. In *BMVC*, 2017. 7

[14] Yuchao Gu, Lijuan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. DOTS: decoupling operation and topology in differentiable architecture search. In *CVPR*, 2021. 1

[15] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 3

[16] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *CVPR*, 2020. 7

[17] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *ICCV*, 2019. 4

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[19] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. DARTS+: improved differentiable architecture search with early stopping. *CoRR*, abs/1909.06035, 2019. 1, 3

[20] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *CVPR*, 2020. 7

[21] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 3

[22] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019. 1, 2, 3, 4, 5, 7

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 1, 2, 3, 7, 8

[24] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *CVPR*, 2018. 5

[25] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 1

[26] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017. 7

[27] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka I. Leon-Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017. 3

[28] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 3

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 3

[30] Yunjie Tian, Chang Liu, Lingxi Xie, Jianbin Jiao, and Qixiang Ye. Discretization-aware architecture search. *Pattern Recognit.*, 2021. 1, 2, 3, 5

[31] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *CVPR*, 2020. 3

[32] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *ICLR*, 2021. 3

[33] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 3

[34] Huikai Wu, Junge Zhang, and Kaiqi Huang. Sparsemask: Differentiable connectivity learning for dense image prediction. In *ICCV*, 2019. 7

[35] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 3

[36] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 3, 4

[37] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. CARS: continuous evolution for efficient neural architecture search. In *CVPR*, 2020. 3

[38] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. β-darts: Beta-decay regularization for differentiable architecture search. In *CVPR*, 2022. 3

[39] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020. 7

[40] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1, 3, 8

[41] Miao Zhang, Wei Huang, and Bin Yang. Interpreting operation selection in differentiable architecture search: A perspective from influence-directed explanations. In *NIPS*, 2022. 3

[42] Xiong Zhang, Hongmin Xu, Hong Mo, Jianchao Tan, Cheng Yang, Lei Wang, and Wenqi Ren. DCNAS: densely connected neural architecture search for semantic image segmentation. In *CVPR*, 2021. 2, 3, 4, 5, 6, 7

[43] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, 2019. 7

[44] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 7

[45] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018. 3

[46] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[47] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 3